📘 Technical Writing Portfolio

Akhilesh Bhagwani – Documentation Samples

Welcome to my technical writing portfolio. The following samples demonstrate my ability to create clear, concise, and developer-centric documentation for complex UI components and controls. Each sample highlights my approach to explaining technical functionality, use cases, and customization options for software developers.

🔷 Tablix Control Documentation

Overview:
The **Tablix** data region is a powerful reporting component that enables developers to arrange and display data in rows and columns, ranging from basic tables to advanced matrices.

Target Audience: Developers building interactive reports with advanced grouping and layout flexibility.

🔷 DashboardLayout Control Documentation

Overview:
The **DashboardLayout** control provides a flexible layout system for creating visually rich and interactive dashboards. It enables the arrangement of visual components such as grids, charts, and maps in various layouts for effective data presentation.

Target Audience: Developers working on data visualization interfaces requiring custom dashboard designs.

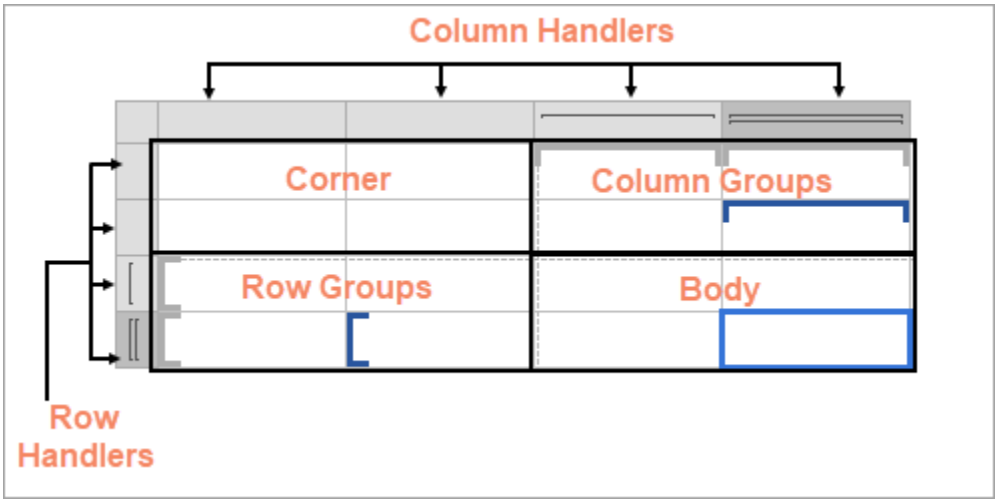
Tablix

A Tablix data region displays data in cells that are arranged in rows and columns. It provides enhanced layout capabilities ranging from the creation of simple tables to advanced matrices.

Tablix is essentially a combination of two data regions, the table, and the matrix. Therefore, it provides all the features of a table and a matrix along with added capabilities including support for multiple adjacent groups on rows or columns and improved layout flexibility with stepped group layouts.

By default, each Tablix cell contains a **TextBox** control, and the function for each cell is determined by its location. You can change the layout of the Tablix data region using the **LayoutDirection** property.

Structure



The Tablix data region is composed of four areas denoted by dotted lines on the design surface - the corner, the row group area, the column group area, and the body.

Corner

The Corner element is located in the upper-left corner, or upper-right corner if a Tablix has the **LayoutDirection** property set to Rtl. The layout direction applies at preview time only. This area is automatically expanded horizontally or vertically when you add a new column or row groups. You can merge cells

inside the Tablix corner and insert a data visualizer such as a TextBox or Image.

A corner may contain only **static cells** that are rendered only once in Tablix.

Column Group

A Column group is represented by square brackets above the columns. Tablix column groups are located in the upper-right corner (upper-left corner for the Rtl layout). A column group represents a member of the column groups hierarchy and displays the column group instance values.

Row Group

A Row group is represented by square brackets on the left side of the rows. Tablix row groups are located on the lower-left corner (lower right for the Rtl layout). A row group represents a member of the row groups hierarchy and displays row group instance values.

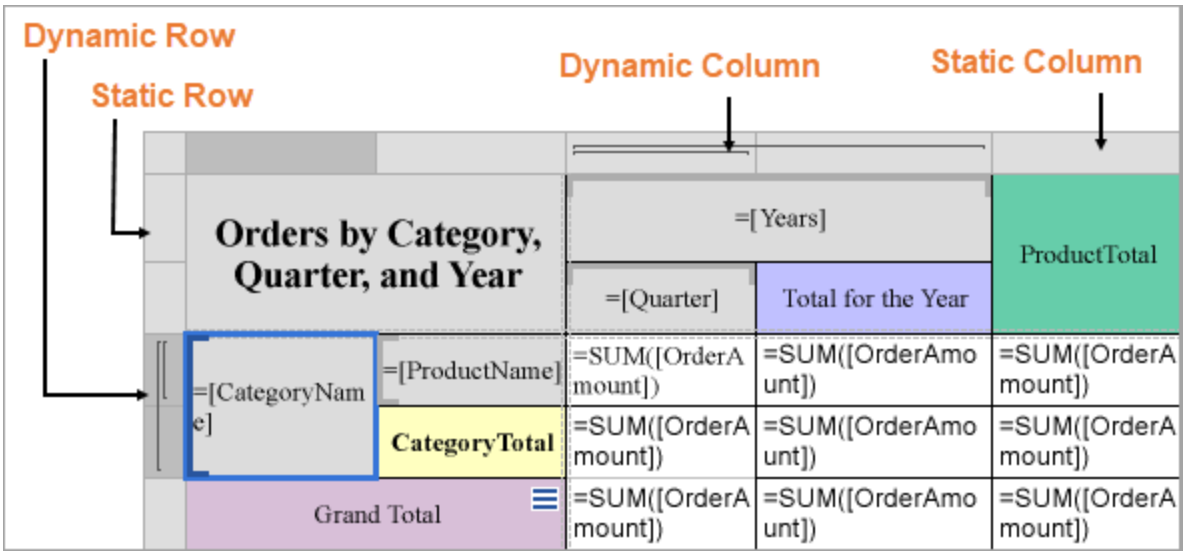
Body

The Body element is used for displaying aggregated data with respect to row and column grouped data in the report. The body may contain only **static cells** that are rendered only once in Tablix.

Static and Dynamic Rows and Columns

Rows or columns in the Tablix data region can be **static** or **dynamic**. The Tablix data region contains multiple rows and columns that provide a grid-type layout, where you can add or remove static or dynamic rows and columns in order to display your data efficiently.

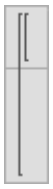
- **Static Rows and Columns** - A static row or column is not associated with any group data. When the report runs, a static row or column is rendered only once. Labels and totals are displayed using static rows or columns in the Tablix data region.
- **Dynamic Rows and Columns** - A dynamic row or column is associated with one or more groups and renders once for every unique value in the group. You can also create dynamic group rows or columns by adding a row group or a column group.



Row and Column Handlers

When you select a Tablix data region, the row and the column handles appear. These handles help you to work with a Tablix and visually specify the type of data added in your Tablix layout.

The following table shows the different types of handles that appear in a Tablix data region.

Handle Icon	Description
	Row or column with one outer group.
	One outer group and one inner group.
	One outer group with an extra row for totals and one inner group.

Tablix Layout Actions

The Tablix data region provides context menu options to perform basic layout actions. You can access layout options for Tablix rows from the context menu by right-clicking on a selected row.

- **Insert Row:** Select from the following options to insert a row inside or outside of the selected group cell.
    - **Inside Group:** If a row group contains groups having distinct values, then as many rows are inserted as there are groups.
        - **Above**: Inserts a row above for each unique value of the row group.
        - **Below**: Inserts a row below for each unique value of the row group.
    - **Outside Group**: If a row group contains nested groups consisting of child and parent groups, then as many rows are inserted as there are parent groups.
        - **Above**: Inserts a row above for each unique value of the parent row group.
        - **Below**: Inserts a row below for each unique value of the parent row group.
- **Delete Row:** Delete the selected rows.
- **Distribute Rows Evenly**: Set the same height for multiple selected rows.
- **Add Row Group:** Select from the following options to insert row groups in a tablix.
    - **Parent Group**: To insert a parent row group.
    - **Child Group**: To insert a child row group.
    - **Adjacent Above**: To insert an adjacent row group above the selected row group.
    - **Adjacent Below**: To insert an adjacent row group below the selected row group.
- **Row Group:** Select the Delete Group option to delete a row group.

You can access layout options for Tablix columns from the context menu by right-clicking on a selected column.

- **Insert Column**: Select from the following options to insert a column inside or outside of the selected group cell.

    - **Inside Group**: If a column group contains groups having distinct values, then as many columns are inserted as there are groups.
        - **Left**: Inserts a column to the left for each unique value of the column group.
        - **Right**: Inserts a column to the right for each unique value of the column group.
    - **Outside Group**: If a column group contains nested groups consisting of child and parent groups, then as many columns are inserted as there are parent groups.
        - **Left**: Inserts a column to the left for each unique value of the parent column group.
        - **Right**: Inserts a column to the right for each unique value of the parent column group.
- **Delete Column**: Delete the selected columns.
- **Distribute Columns Evenly:** Set the same width for multiple selected columns.
- **Add Column Group:** Select from the following options to insert column groups in a tablix.
    - **Parent Group**: To insert a parent column group.
    - **Child Group**: To insert a child column group.
    - **Adjacent Left**: To insert an adjacent column group to the left of the selected column group.
    - **Adjacent Right**: To insert an adjacent column group to the right of the selected column group.
- **Column Group:** Select the Delete Group option to delete a column group.

Group Editor

The Group Editor window gets displayed or hidden by the Group Editor icon, located on the left of the Design area. The Group Editor contains the following groups:

- **Row Groups:** The Row Groups section in the Group Editor displays all the groups that are applied in the row group area of the Tablix data region.
  Category Name
  The left-center cell of the Tablix data region **=[CategoryName]** represents the **Category** group in the Group Editor window. This group displays the category names for products.
  Product Name
  The left-center cell of the Tablix data

region **=[ProductName]** represents the **Product** group in the Group Editor window. This group displays the product names.

- **Column Groups:** The Column Groups section in the Group Editor window displays all the groups that are applied in the column group area of the Tablix data region.
  Year
  The center-left cell of the Tablix data region **=[Years]** represents the **Years** group in the Group Editor window. This group displays years of orders.
  Quarter
  The center-right cell of the Tablix data region **=[Quarter]** represents the **Quarter** group in the Group Editor window. This group displays quarters for the orders.
- **Static Cells:** Static cells in the Row Groups and Column Groups are not represented in the Group Editor window because these cells are not associated with any grouped data. Static row and column cells are used to display labels and totals in a Tablix data region.

The static column cell displays the label **YEARS** and **CATEGORY NAMES** in the Tablix data region. The static row cell displays the label **Total** in the Tablix data region.

The image below displays the subjects in a row group. Nested column groups display practical and theory scores for the students. The total row displays the total scores for all of the subjects.

Orders by Category, Quarter, and Year		1994			1995		
		Q 3	Q 4	Total for the Year	Q 1	Q 2	Q 3
Grains/Cereals	Singaporean Hokkien Fried	$98,00	$302,40	$400,40	$448,00	$1 484,00	$1 862,00
	Gustaf's Knäckebröd	$100,80		$100,80	$201,60	$504,00	$3 003,00
	Ravioli Angelo	$1 045,20	$1 029,60	$2 074,80	$546,00	$97,50	$585,00
	Gnocchi di nonna Alice	$60,80	$1 702,40	$1 763,20	$6 870,40	$8 177,60	$8 626,00
	Wimmers gute Semmelknödel	$239,40	$2 261,00	$2 500,40	$2 872,80	$1 010,80	$2 094,75
	Filo Mix		$156,80	$156,80	$308,00	$42,00	$931,00
	Tunnbröd		$468,00	$468,00	$720,00	$1 141,20	$90,00
	CategoryTotal	$1 544,20	$5 920,20	$7 464,40	$11 966,80	$12 457,10	$17 191,75
Dairy Products	Mozzarella di Giovanni	$1 786,40	$3 058,00	$4 844,40	$3 808,60	$2 609,00	$4 906,80
	Queso Cabrales	$168,00	$1 646,40	$1 814,40	$1 209,60	$2 457,00	$1 554,00
	Geitost	$258,00	$16,00	$274,00	$398,00	$196,50	$170,00
	Camembert Pierrot	$3 155,20	$3 889,60	$7 044,80	$4 651,20	$6 426,00	$6 086,00

Tablix Features

Column and Row Groups

Groups categorize the report data using a specified expression. You can add a group by using the context menu options or in the Group Editor component that is part of the designer.

In Tablix, you can add row/column groups in the following ways:

- **Parent-child groups**: To depict the hierarchical relation.
- **Adjacent groups**: To show the side-by-side grouping of report data.

Important Group and Layout Properties for Tablix Groups

- **Filters**: The filters to apply on the group.
- **GroupExpressions**: Specifies the group expression for the group.
- **NewSection**: Determines whether each group instance has its page numbering.

- **PageBreak**: Indicates how the rendering engine inserts a page break in relation to the group.
    - **BreakLocation:** Determines the location of page breaks generated by group instances.
        - None: no page breaks are generated.
        - Start: each group instance inserts the page break before printing its content.
        - End: each group instance inserts the page break after printing its content.
        - StartAndEnd: the combination of the Start and End options.
        - Between: each group instance starts on the new page.
    - **Disabled**: Indicates whether the page break properties should be ignored. The expression allows you to conditionally prevent page breaks from being inserted by the aforementioned property.
    - **NewPage**: Indicates on which page the content to start after the page break.
        - **Next**: A default value that makes a new group start from the immediate next page of the report.
        - **Odd**: A new group starts from the next odd page of the report.
        - **Even**: A new group starts from the next even page of the report.

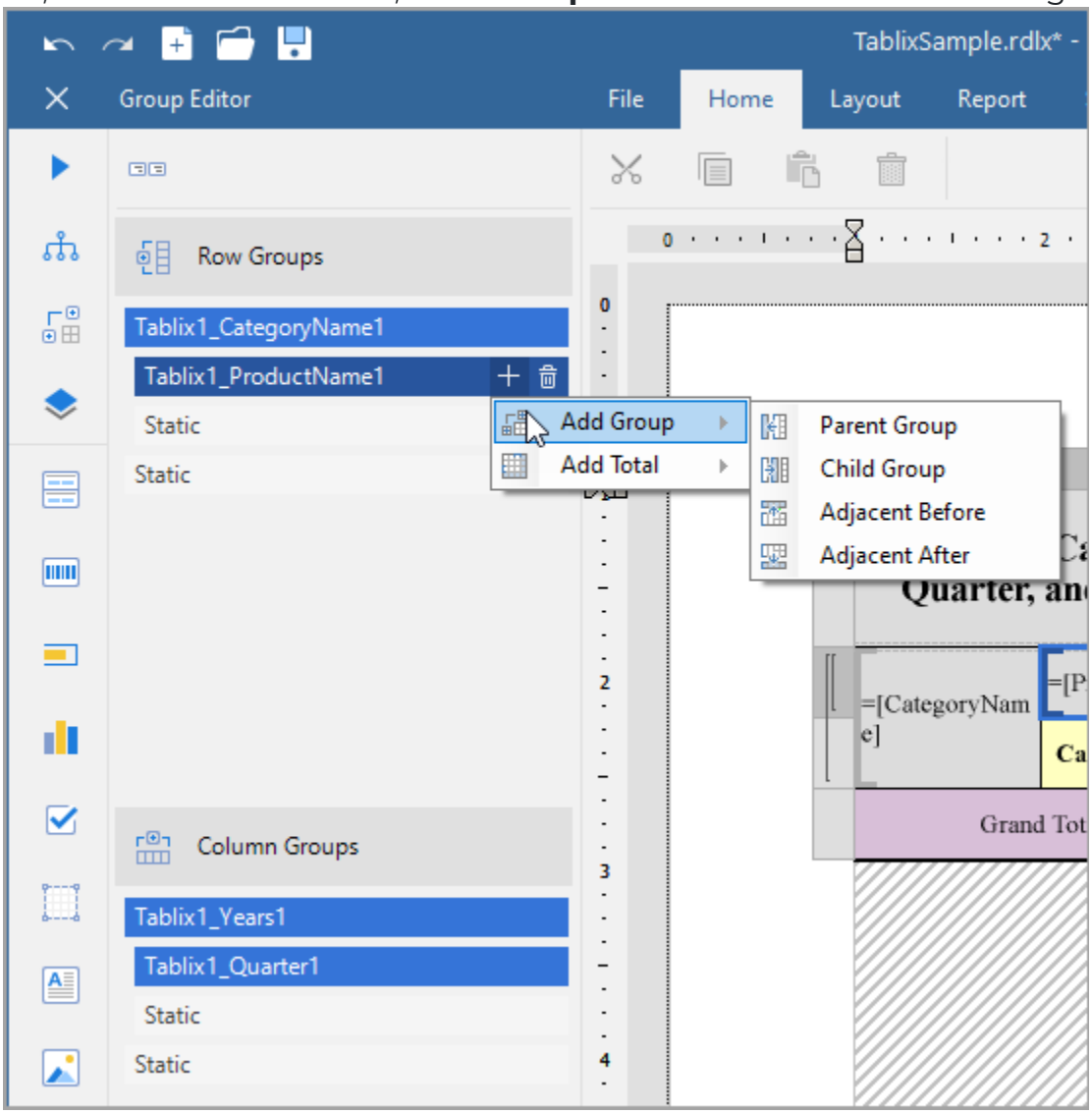
  See Manage Page Breaks in Data Regions topic for more detail.

- **KeepTogether**: Setting it True ensures that the group instance always appears on a single page if it fits.

To add a group

1. Select the Tablix data region in the design area.
2. Right-click and select any group action in the context menu that appears.

3. Or, with a Tablix selected, click **Group Editor** on the left of the Designer.



Totals

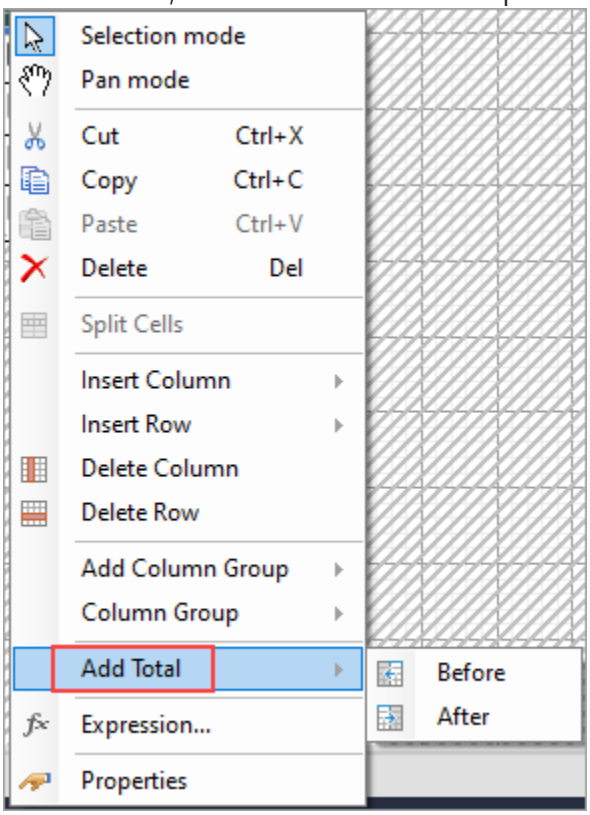
The intersection of a Row and Column Group displays one or more summary values. For example, in a report output, Row Groups may display **media types**, Column Groups - **quarters**, and their intersections may show the **Sales** for each media type in each quarter.

A tablix can display two types of totals.

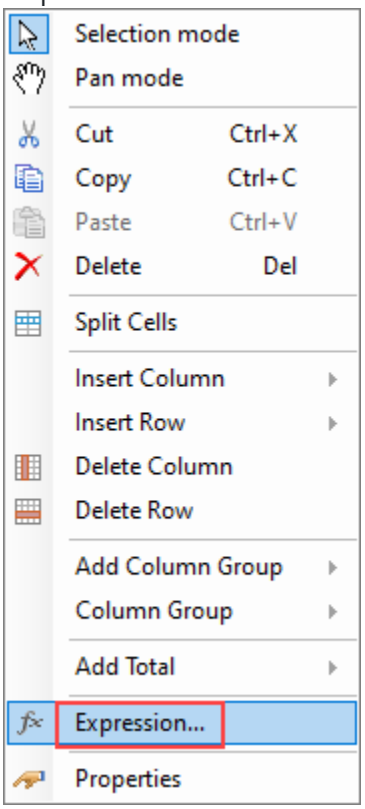
- The **Grand Total** appears at the beginning or at the end of all the group instances.
- The **Subtotal** appears at the beginning or at the end of each group instance.

To add a total,

1. Right-click a Tablix cell and select **Add Total > Before** or **Add Total > After** menu item.
   If a group does not have a parent, then the grand total will be inserted. Otherwise, the subtotal for the parent group will be created.



2. To insert additional totals, you can add rows and columns and set an expression to calculate sum for the newly added cells.



Merge Cells

Tablix cells with the same value in a row group or a column group area are automatically merged. In the case of static cells, you can combine adjacent cells in horizontal (same row) or vertical (same column) direction into a single cell. For example, you may want a column header to span across the columns.

To merge cells,

1. Select the cells (Ctrl+Click) and then right-click.
2. From the context menu, select **Merge Cells**.

Auto Merge

The **AutoMergeMode** property lets you set the mode to merge the adjacent cells (text boxes) in a row group with the same value. This property takes Always, Never, and Restricted values. The row groups with the same data values and with AutoMergeMode property set to:

- **Always** - are merged.
- **Never** - are not merged.
- **Restricted** - are merged only if the corresponding cells in the previous columns are similarly merged. If for example, cells in Column 2 (with the same data values) are set 'Restricted' and the corresponding cells (with the same data values) in previous column, that is Column 1, are set 'Never', then cells in Column 2 are not merged.

Let us take an example. In the following simple Tablix data region, 'District' column is **Outside Group - Right** to the 'Region' column group. We want the 'District' values to merge in case the 'Region' values are the same.
The tablix without AutoMergeMode set in any cell looks as follows:

Store Name	Region	District
HQ	No Region	No District
Store #1000	North West	Portland
Store #1001	Canada West	Victoria
Store #1002	North West	Seattle
Store #1003	Canada West	Vancouver
Store #1004	Canada West	Vancouver
Store #1005	Canada West	Victoria
Store #1006	North West	Portland
Store #1007	Canada West	Vancouver

Let us set the **AutoMergeMode** property for the 'District' and the 'Region' values.

1. Select the cell with `District` field in the row group and set the **Layout > AutoMergeMode** property to 'Restricted'. This merges districts depending on whether the corresponding regions (cells in the previous column) are merged.
2. Select the cell with `Region` field in the row group and set the **Layout > AutoMergeMode** property to 'Always'. This merges the cells with similar regions.

3. Preview the report. Here's how the tablix will look like.

Store Name	Region	District
HQ	No Region	No District
Store #1000	North West	Portland
Store #1001	Canada West	Victoria
Store #1002	North West	Seattle
Store #1003	Canada West	Vancouver
Store #1004		
Store #1005		Victoria
Store #1006	North West	Portland
Store #1007	Canada West	Vancouver

Repeat to Fill (Page report)

If you set the **RepeatToFill** property to True (the default value is False), the Tablix will fill extra space with empty rows.

Setting the **RepeatToFill** property to 'True' (default value is 'False') fills the tablix with empty rows to reach the Fixed Height of the tablix. So, each page of the report displays the tablix with the same height. For example, the following image shows the page of a report where **RepeatToFill** property for the Tablix data region is set to 'False':

USA	Beaverton	DVD	$563,08	$854,94	$1 033,46	$481,12	$2 932,60
		HD-DVD	$64,90	$59,98	$189,74	$33,95	$348,57
		LaserDisc	$403,85	$786,90	$583,61	$641,59	$2 415,95
		VHS	$714,17	$775,68	$948,34	$475,46	$2 913,65
		City Total	$1 746,00	$2 477,50	$2 755,15	$1 632,12	$8 610,77
	Issaquah	DVD	$335,65	$1 104,47	$650,30	$1 848,63	$3 939,05
		HD-DVD		$100,93	$35,99	$85,85	$222,77
		LaserDisc	$209,62	$961,01	$591,73	$801,48	$2 563,84
		VHS	$389,01	$1 024,18	$445,57	$1 284,09	$3 142,85
		City Total	$934,28	$3 190,59	$1 723,59	$4 020,05	$9 868,51
	W. Linn	DVD	$515,38	$864,10	$930,74	$914,54	$3 224,76
		HD-DVD	$29,95	$166,83	$33,95	$115,84	$346,57
		LaserDisc	$661,53	$457,24	$980,70	$649,71	$2 749,18
		VHS	$944,26	$1 133,93	$822,60	$955,62	$3 856,41
		City Total	$2 151,12	$2 622,10	$2 767,99	$2 635,71	$10 176,92
	Country Total		$4 831,40	$8 290,19	$7 246,73	$8 287,88	$28 656,20
Total			$12 492,68	$14 783,74	$15 768,20	$18 266,29	$61 310,91

Then, in the Properties panel, set the **RepeatToFill** property to 'True'. At the report preview, you will see that the tablix now has additional empty rows on the same page of the report.

USA	Beaverton	DVD	$563,08	$854,94	$1 033,46	$481,12	$2 932,60
		HD-DVD	$64,90	$59,98	$189,74	$33,95	$348,57
		LaserDisc	$403,85	$786,90	$583,61	$641,59	$2 415,95
		VHS	$714,17	$775,68	$948,34	$475,46	$2 913,65
		City Total	$1 746,00	$2 477,50	$2 755,15	$1 632,12	$8 610,77
	Issaquah	DVD	$335,65	$1 104,47	$650,30	$1 848,63	$3 939,05
		HD-DVD		$100,93	$35,99	$85,85	$222,77
		LaserDisc	$209,62	$961,01	$591,73	$801,48	$2 563,84
		VHS	$389,01	$1 024,18	$445,57	$1 284,09	$3 142,85
		City Total	$934,28	$3 190,59	$1 723,59	$4 020,05	$9 868,51
	W. Linn	DVD	$515,38	$864,10	$930,74	$914,54	$3 224,76
		HD-DVD	$29,95	$166,83	$33,95	$115,84	$346,57
		LaserDisc	$661,53	$457,24	$980,70	$649,71	$2 749,18
		VHS	$944,26	$1 133,93	$822,60	$955,62	$3 856,41
		City Total	$2 151,12	$2 622,10	$2 767,99	$2 635,71	$10 176,92
	Country Total		$4 831,40	$8 290,19	$7 246,73	$8 287,88	$28 656,20

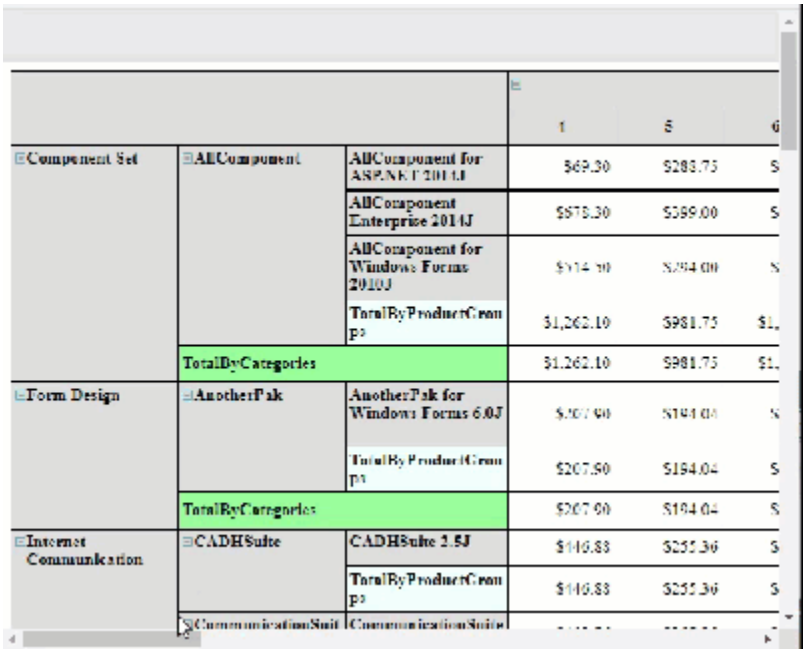
Freeze Rows and Columns (RDLX Reports)

When you use a Tablix data region containing a large amount of data in an RDLX report, the user must scroll to see all of the data. On scrolling the column or row headers out of sight, the data becomes difficult to understand. To resolve this problem, you can use the **FrozenRows** and **FrozenColumns** properties that take effect in the JSViewer in Galley mode, and allow you to freeze headers so that they remain visible while scrolling through the data region. You can freeze as many rows or columns as you have headers in the data region.

- If your data stretches downward, set the FrozenRows property to a value to float the column headers when scrolling.
- If your data stretches to the right, set the FrozenColumns property to a value to float the row headers when scrolling.

- If your data stretches both downward and to the right, set both FrozenRows and FrozenColumns properties.

Here's how a report with one frozen row and column headers looks like in JS Viewer.

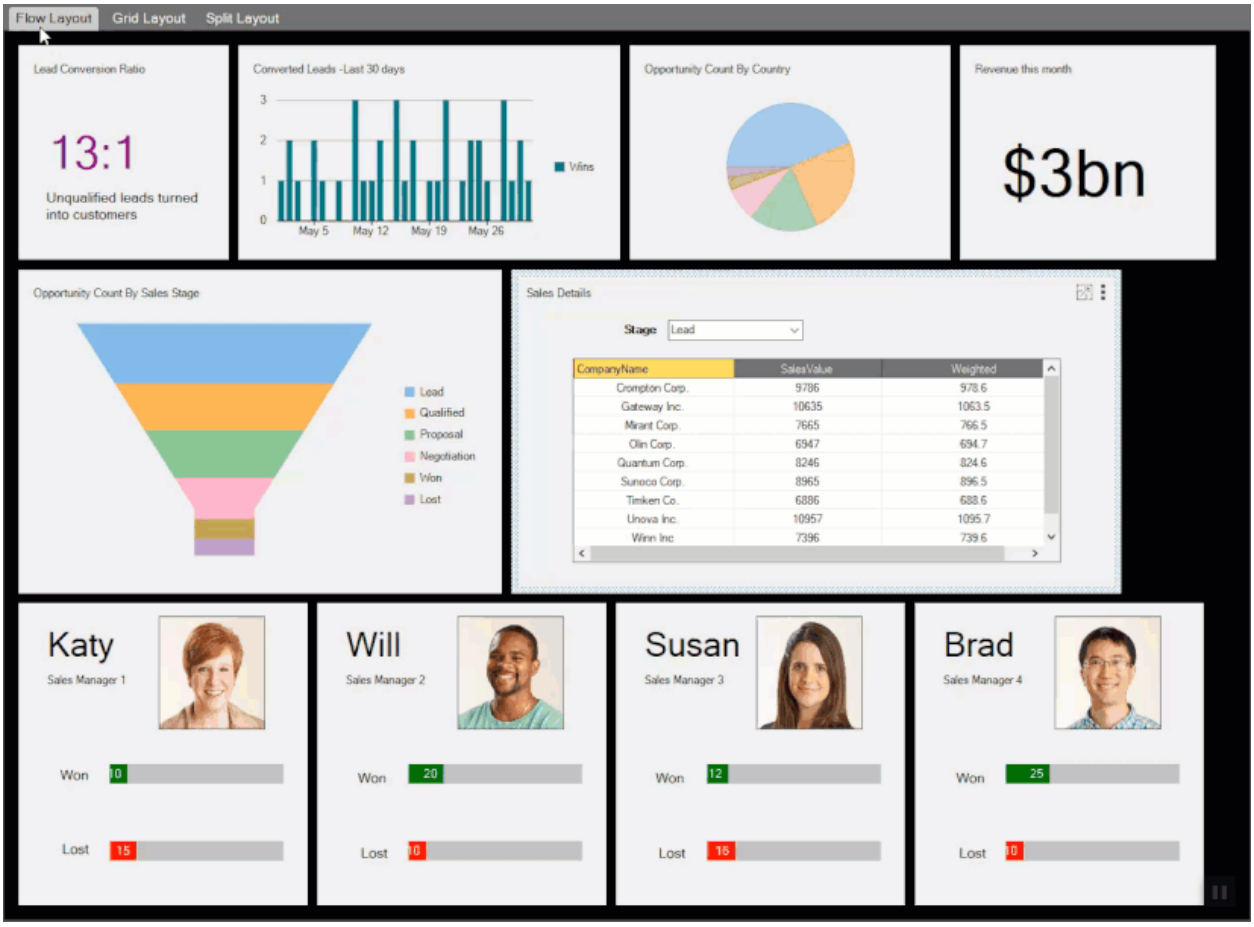


If any header cells that you want to freeze are merged, you should not set the **FrozenRows** or **FrozenColumns** property to a value that would split a merged cell.

DashboardLayout

DashboardLayout is a layout control that allows you to create dynamic dashboards for interactive data visualization. It allows you to organize and present data in a consolidated form with the help of images, grids, charts, maps, etc in different layouts. This makes it easy for you to monitor the presented information.

The DashboardLayout control acts as a container which lets you dynamically place controls within tiles also called child containers. These child containers can be arranged in four different types of layouts supported by the DashboardLayout control i.e. Flow, AutoGrid, ManualGrid and Split. One of these layouts is attached to the DashboardLayout control to achieve the desired result. The control lets you resize and rearrange these child containers at runtime to create an ideal workspace.



Key Features

DashboardLayout provides various features that enable the developers to build intuitive and professional-looking dashboards. The main features for DashboardLayout are as follows:

- **Use different layouts**
  DashboardLayout provides four types of layouts, namely Flow, AutoGrid, ManualGrid and Split. These layouts specify the arrangement of the tiles in different ways on the DashboardLayout control.

- **Tile headers**
  DashboardLayout control provides a header for each tile, which displays a caption and provides options to maximize/restore or show/hide the tiles.

- **Drag and drop tiles**
  DashboardLayout allows you to drag and drop a tile using two methods; crosshair icon and tool-icon. The crosshair appears when you hover the mouse over the tile header. As an alternative, you can also use the tool-icon to perform the drag drop operation by holding the cursor on the tool-icon.

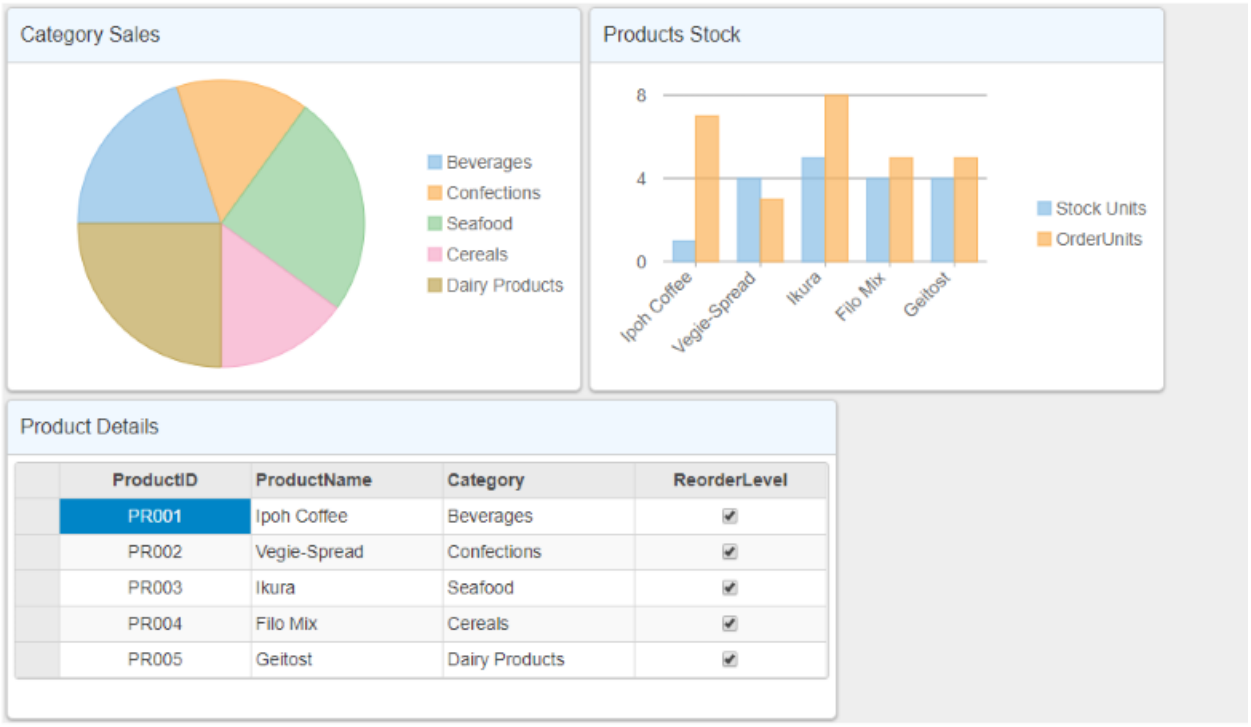
- **Maximize/Restore Tiles**
  DashboardLayout control's header provides a maximize/restore icon, which is used to maximize or restore the display state of the tile.

- **Perform multiple functions with ToolIcon**
  DashboardLayout provides a multi functional tool called Tool-icon, which appears as an icon in the header or on the surface of the tile on hovering the mouse over it. It can be used to drag and drop the tiles. Moreover, tool-icon provides a context menu with options to hide a tile or show all the tiles.

- **Serialization/Deserialization**
  DashboardLayout offers XML serialization by providing the functionality to save the layout properties of the control to an XML file or stream and load them from another XML file or stream.

Quick Start

The quick start guides you through the steps of adding DashboardLayout control to your MVC web application for creating a simple dashboard application. Follow the steps given below to get started:



Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Add Data to the Application

1. Add a new class to the **Models** folder (Name: `ProductDashboardData.cs`). For more information on how to add a new model, see Adding Controls.
2. Add the following code to `ProductDashboardData.cs` model. We are using `ProductDashboardData` class to represent data.

C# Copy Code

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Dashboard_Quickstart.Models
{
    public class ProductDashboardData
    {
        private IEnumerable<ProductData> _productDetails = null;
        public IEnumerable<ProductData> ProductDetails
        {
            get
            {
                if (_productDetails == null)
                {
                    _productDetails = GetProductData();
                }
                return _productDetails;
            }
        }
        public IEnumerable<ProductData> GetProductData()
        {
            var rand = new Random(0);
            var productID = new[] { "PR001", "PR002", "PR003", "PR004",
"PR005" };
            var products = new[] { "Ipoh Coffee", "Vegie-Spread", "Ikura",
"Filo Mix",
        "Geitost" };
            var categories = new[] { "Beverages", "Confections", "Seafood",
"Cereals",
        "Dairy Products" };

            var list = products.Select((p, i) =>
            {
                int stockunits = rand.Next(1, 6);
                int orderunits = rand.Next(1, 9);
                int sales = rand.Next(1, 6);
```

```
                return new ProductData { ProductID = productID[i], ProductName
= p,
        Category = categories[i], UnitsInStock = stockunits, UnitsOnOrder =
orderunits,
        Sales = sales, ReorderLevel = true };
            });

            return list;
        }
    }
    public class ProductData
    {
        public string ProductID { get; set; }
        public string ProductName { get; set; }
        public string Category { get; set; }
        public int UnitsInStock { get; set; }
        public int UnitsOnOrder { get; set; }
        public int Sales { get; set; }
        public bool ReorderLevel { get; set; }
    }
}
```

Add a DashboardLayout Control

Steps to add a DashboardLayout control to the application, are as follows:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template, and then click **Add**.
    2. Set name of the controller (for example: `DashboardLayoutController`).
    3. Click **Add**.
4. Include the following references as shown below.

C# Copy Code

```
using <ApplicationName>.Models;
```

5. Replace the **Index()** method with the following method.

DashboardLayoutController.cs Copy Code

```
public ActionResult Index()
        {
            ProductDashboardData data = new ProductDashboardData();
            return View(data.ProductDetails);
        }
```

Add a View for the Controller

In the view, we create an instance of **DashboardLayout** and attach the flow layout to the control by using the **AttachFlowLayout** method provided by the **DashboardLayoutBuilder** class.

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `DashboardLayoutController.`
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller, and then copy the following code and paste it inside **Index.cshtml**.

Index.cshtml Copy Code

```
@model IEnumerable<ProductData>


<style>
    .wj-dashboard .wj-flexchart {
```

```
        margin: 0px;
        padding: 4px;
        border: none;
        height: 240px;
    }
</style>
<c1-dashboard-layout id="SampleDashboard" allow-drag="true" allow-hide="true"
        allow-maximize="true" allow-resize="true" allow-show-all="true">
    <c1-flow-layout direction="LeftToRight">
        <c1-flow-tile header-text="Category Sales" width="450" height="300">
            <c1-flex-pie binding-name="Category" binding="Sales">
                <c1-items-source source-collection="Model"></c1-items-source>
            </c1-flex-pie>
        </c1-flow-tile>
        <c1-flow-tile header-text="Products Stock" width="450" height="300">
            <c1-flex-chart binding-x="ProductName" legend-position="Top"
chart-type="Column">
                <c1-items-source source-collection="@Model"></c1-items-source>
                <c1-flex-chart-series name="Stock Units"
binding="UnitsInStock">
        </c1-flex-chart-series>
                <c1-flex-chart-series name="Order Units"
binding="UnitsOnOrder">
        </c1-flex-chart-series>
            </c1-flex-chart>
        </c1-flow-tile>
        <c1-flow-tile header-text="Product Details" width="650" height="250">
            <c1-flex-grid auto-generate-columns="false">
                <c1-flex-grid-column binding="ProductID" width="150"></c1-
flex-grid-column>
                <c1-flex-grid-column binding="ProductName" width="150"></c1-
flex-grid-column>
                <c1-flex-grid-column binding="Category" width="150"></c1-flex-
grid-column>
                <c1-flex-grid-column binding="ReorderLevel" width="150"></c1-
flex-grid-column>
```

```
                <c1-items-source source-collection="@Model" width="150"></c1-
items-source>
            </c1-flex-grid>
        </c1-flow-tile>
    </c1-flow-layout>
</c1-dashboard-layout>
```

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

Append the folder name and view name to the generated URL (for example: http://localhost:1234/**DashboardLayout/Index**) in the address bar of the browser to see the view.